Internet Programming in Python - Week 3

Mashups and web APIs

Brian Dorsey
brian@dorseys.org

http://www.delicious.com/briandorsey/uwpython2010+week03

http://www.delicious.com/tag/uwpython2010+week03

A moment to reflect

1

2

6

24

120

720

The Evolution of a Python Programmer

http://metaleks.net/programming/the-evolution-of-a-python-programmer

fastest

```
def factorial(x):
    result = 1
    i = 2
    while i <= x:
        result = result * i
        i = i + 1
    return result
print factorial(n)
```

https://gist.github.com/788698

```
import math
math.factorial(6)
```

http://svn.python.org/view/python/trunk/Modules/mathmodule.c?view=markup

Questions and review (20)

some thoughts from the assignments

Assignment

- Update thirty_minute_webserver.py
- Instead of showing the content of .py files, run the script and return its' output instead.
(the stdlib subprocess module will be useful)
- test with script which outputs the time (print_time.py)
- update the print_time.py script to output html
- You've now written a dynamic webserver!
- Turn in: http://bit.ly/upipip_week2

most common

```
# somewhere in get_content


process = subprocess.Popen(['python',path],
                                    stdout=subprocess.PIPE)
return process.communicate()[0]
```

```
c = subprocess.Popen('python ' + path,
                     stdout=subprocess.PIPE, shell=True)
c.wait()
pyProcess = c.stdout.read()
return (200, get_mime(uri), pyProcess)
```

```
command2run = ['python',inFile]
rVal = subprocess.Popen(command2run,
                  stdout=subprocess.PIPE,
                  stderr=subprocess.PIPE).communicate()
#test to make sure return value did not error
if rVal[1] != "":
     rVal = rVal[1]
else:
     rVal = rVal[0]
```

```
child_proc = subprocess.Popen('python '+path,
             shell=True, stdout=subprocess.PIPE)
lst = []
# collect all output to stdout into a list
for line in child_proc.stdout:
    print '[parent_proc] '+ line.strip()
    lst.append(line)
if re.match('<html>', lst[0]):
    # swap MIME type to html
    m_type = 'text/html'
else:
    # swap MIME type to plain text
    m_type = 'text/plain'
return (200, m_type, ''.join(lst))
```

```
if uri.endswith('.py'):
    print 'Runnying Python file.'
    return (200, 'text/html', run_script(path))


def run_script(path):
    process = subprocess.Popen(['python',path],
stdout=subprocess.PIPE)
    return process.communicate()[0]
```

```
# in get_file:
def get_file(path):
    if path.endswith('.py'):
        return subprocess.Popen(['python', str(path)],
                shell=False,
                stdout=subprocess.PIPE).communicate()[0]
```

VARIATION

- `get_content()` vs. `get_file()`
- function for executing the script
- error handling

questions?

- hard coded script names (or URLs) instead of running any .py file

- PIPE is only so big, you may need to iterate with communicate, or use a tempfile instead.

- didn't see anyone using **`returncode`** (error level in Windows)

Lecture A (20)

Web Scraping

HTTP

HTML

```
<html>
    <head>
    </head>
    <body>
        <p>Here is the time: %s</p>
        <p>and again: %s</p>
    </body>
</html>
```

Robustness principle

"Be liberal in what you accept, and conservative in what you send."

-- Jon Postel

http://postel.org/postel.html

this was originally referring to TCP implementations, but has had a wide influence on internet protocols at all levels

real world html dials it to 11

```
<html>
<form>
 <table>
 <td><input name="input1">Row 1 cell 1
 <tr><td>Row 2 cell 1
 </form>
 <td>Row 2 cell 2<br>This</br> sure is a long cell
</body>
</html>
```

http://www.crummy.com/software/BeautifulSoup/documentation.html

What is webscraping?

my take:
software browsing the web
instead of humans

```
import urllib2

data = urllib2.urlopen('http://briandorsey.info')
print data.read()
```

now what?

```
<html>
<head>
    <meta name="verify-v1" content="5T6/TWLAsZrBIZHND4LZ/rWuxcSkCyXfvW7GiQgqA2Q=" />
    <meta name="verify-v1" content="XJ0W1/sed65X/hYXVLdcMtWqF5+gVxKks+hDUwaAQHY=" />
    <!-- claimid stuff -->
    <link rel="openid.server" href="http://openid.claimid.com/server" />
    <link rel="openid.delegate" href="http://openid.claimid.com/briandorsey" />
    <meta name="microid" content="mailto+http:sha1:e5962a404b32f2df2296faac8b40569b07e17433" />
    <!-- end claimid stuff -->
    <link rel="shortcut icon" href="./favicon.jpg"/>
    <link rel="apple-touch-icon" href="./favicon.jpg"/>
<title>
Brian Dorsey
</title>
</head>
<body>
<br><br>
<table>
    <tr>
        <td>
            <img src="Brian_Dorsey_Bio_Photo_200.jpg" alt="Photo of Brian Dorsey" width=200 height=200>
        </td>
        <td>
        </td>
        <td>
            Brian Dorsey<br>
            Seattle, Washington, USA<br>
            <br>
            twitter: <a href="http://twitter.com/briandorsey/">@briandorsey</a><br>
            blog: <a href="http://blog.briandorsey.info/">http://blog.briandorsey.info/</a><br>
            email: brian@dorseys.org<br>
            cell: 206-569-4371<br>
            <br>
            Take care,<br>
              -Brian<br>
            <br>
        </td>
    </tr>
</table>
<a href="http://www.twitter.com/briandorsey"><img src="http://twitter-badges.s3.amazonaws.com/t_mini-a.png" alt="Brian on Twitter" title="Twitter" border=none /></a>
<a href="http://www.flickr.com/photos/briandorsey"><img src="http://www.flickr.com/favicon.ico" alt="Brian on Flickr" title="Flickr" border=none /></a>

<p><strong>Active projects:</strong>
<ul>
    <li><a href="/uwpython/">UW Internet Programming in Python</a> (10 week evening course)</li>
</ul>

<p><strong>Recent projects:</strong>
<ul>
    <li><a href="http://www.noonhat.com">Noonhat</a> - a website to introduce you to random other people for lunch conversation.</li>
</ul>
</p>

<link rel="meta" type="application/rdf+xml" title="FOAF" href="foaf.rdf" />

<script src="http://www.google-analytics.com/urchin.js" type="text/javascript">
</script>
<script type="text/javascript">
_uacct = "UA-1707974-2";
urchinTracker();
</script>
</body>
```

string methods

regular expressions

Some people, when confronted with a problem, think

"I know, I'll use regular expressions."

Now they have two problems.

--Jamie Zawinski

stdlib module: HTMLParser

lxml

BeautifulSoup

BeautifulSoup example code

`lab_a_example_google_spell.py`

http://seapig.org/JobsPage

Knowledge Mosaic Corporation

Posted: January 14, 2011

Calling all python programmers!

We are looking for talented python programmers to help with web harvesting. If you understand the web, html, xml, xpath, regex, and sql and love to write great code, then this project is for you.

To apply, we've got a mini programming task that we need you to complete. A one page description is here: http://www.knowledgemosaic.com/websitelinks/harvesttask.pdf. When you're done, send your source code and the text output file to careers@knowledgemosaic.com along with any information you'd like to share about your past accomplishments and future aspirations. Make sure the source executes (no bugs!). We'll run it to verify it produces the output. If it looks good on our end, you can count on us getting in touch at which time we'd be happy to discuss the project and answer any questions you have.

Knowledge Mosaic is a Seattle based company founded in 2001 and growing steadily ever since. We develop leading edge software and provide a research platform, custom data feeds, and deep analytics that are used by an A-List of top companies, law firms, investment firms, news organizations, and government agencies. We have an awesome software dev team and love what we do.

Aside from knowing python and understanding the principles of web harvesting, you don't need any specific experience to apply. Successfully completing the above task is sufficient.

This is a 2-3 month contract with the potential for extension. We pay well at an hourly rate that depends on your experience and ability. We're open to applicants who want to work part time. But we need to have a commitment of at least 20 focused productive hours each week.

This is a "first come, first serve" opportunity and you can start right away. Write some beautiful code. We're looking forward to hearing back from you.

scraping is inherently brittle, someday the site will change...

and your code will break

questions?

Lab A (20)

groups of 4ish if possible, pairs if not

```
git clone git://github.com/briandorsey/uwpython_web.git
```

Lab A

- Scrape the course syllabus and make a list of every date we have a class.

- BeautifulSoup docs: **http://www.crummy.com/software/BeautifulSoup/documentation.html**

You'll need the docs. It's confusing.

Break

(10)

Guest Speaker

(20 +10)

John Musser

ProgrammableWeb.com

API Talks (20)

talks

Break

(10)

RANDOM

```
from __future__ import print_function
```

```
from __future__ import braces
```

```
import this
```

Lecture B (20)

Structured data via HTTP
XMLRPC, REST, etc

HTML

scraping is tricky and brittle

what if we used a structured data format instead?

anything goes, if you can serialize & deserialize

some common formats

XML

XML is really specification for making more specifications

SOAP, XML-RPC, RSS, ATOM, GDATA, KML, OPML, DOCX, ODT, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC, ETC

JSON

http://www.json.org/

JSON
(JavaScript Object Notation)

http://www.json.org/

```
{'array': [],
 'false': False,
 'null': None,
 'number': 123,
 'number_': 123.321,
 'object': {},
 'string': 'abc',
 'true': True}

{"object": {},
 "false": false,
 "string": "abc",
 "number_": 123.321,
 "array": [],
 "null": null,
 "true": true,
 "number": 123}
```


JSON actually has a data model, it’s minimal, but it exists.

annoyance:
no date type

usually solved with:
unix epoch
ISO 8601

```
>>> time.time()
1295998747.5772631


>>> now = datetime.datetime.now()
>>> now.isoformat()
'2011-01-25T15:40:48.963153'
```

structured data is not enough...

Register / Login

Find APIs, mashups, code and developers | Search

programmableweb

Hot APIs » Twitter YouTube Facebook Google Maps Flickr LinkedIn More »

Latest news Dive Deep into REST at Second Annual Workshop

Home | API News | API Directory | Mashups | Community | How-to | Subscribe:

Web Services Directory

Subscribe to get the latest APIs

Hide Filters

Sort by: Name | Date | Popularity | Category

Keywords | Category | Company | Protocols / Styles

Data Format | Date (All)

Filter This List

Viewing 1 to 2755 of 2755 APIs

API	Description	Category	Updated
OneLogin	Single sign-on solution	Security	2011-03
.tel	Access to .tel DNS	Internet	2010-0
10x10	Photo and news analysis service	Photos	2010-0
123 Shop Pro	Online shopping cart software	Shopping	2010-0
12seconds.tv	12-second videoblogging	Video	2010-0
140 Proof	Twitter advertising service	Advertising	2010-12
18amail	Email marketing service	Email	2010-1
1DayLater	Business expense tracking tool	Enterprise	2008-0
2-WaySMS	SMS messaging service	Messaging	2011-0
23	Photo sharing service	Photo	2006-02-13
2Checkout	Ecommerce services	Shopping	2010-03-24
30 Boxes	Calendar service	Calendar	2006-02-24
30 Digits Web Extractor	Semantic web extraction tool	Internet	2010-11-2
360voice	XBox gamer profile data	Games	2010-1 11
3DArmory	World of Warcraft armory database	Games	2009-08-07
3dCart	Online shopping cart service	Shopping	2010-03-25
3Jam	Reply-all text messaging service	Messaging	2008-09-03
3NGNetworks	VoIP services provider	Telephony	2011-01-14
3scale Networks	API management services	Tools	2009-08-21

- ✓
- AIM (OSCAR)
- Atom
- Blogger
- DTC-XML
- GData
- GET
- hCalendar
- iCal
- JavaScript
- JSON-RPC
- POST
- REST
- RSS
- SMS
- SOAP
- XML-RPC
- XMPP

Become a ProgrammableWeb Sponsor

XML-RPC

XML-RPC

"Does distributed computing have to be any harder than this? I don't think so."

what is it?

```
from DocXMLRPCServer import DocXMLRPCServer

# Create server
server = DocXMLRPCServer(("localhost", 8000))

# Register a function
def echo(message):
    "Accepts a message parameter and returns it."
    return message

server.register_function(echo)

# Run the server's main loop
server.serve_forever()

# visit http://localhost:8000/ to see docs
```

lab_b_example_xmlrpc_echo_server.py

http://tools.ietf.org/html/rfc5321#appendix-D

XML-RPC Server Documentation

This server exports the following methods through the XML-RPC protocol.

Methods

echo(message)

Accepts a message parameter and returns it unchanged.

```
import xmlrpclib

s = xmlrpclib.ServerProxy('http://localhost:8000')

print s.echo('hello')
print s.echo('world')
```

lab_b_example_xmlrpc_echo_client.py

looks pretty much like a local call, right?

REST

originally described in Roy T. Fielding's original dissertation

If you decide to write one, read this.

http://www.crummy.com/writing/RESTful-Web-Services/

"We want to shift the focus of web service programming from an RPC-style architecture that just happens to use HTTP as a transfer protocol, to a URI-based architecture that uses the technologies of the web to their fullest."

http://www.crummy.com/writing/RESTful-Web-Services/

an oversimplification

go with the grain of HTTP

```
import urllib2
import json
from pprint import pprint

# urlopen returns a file-like object
data = urllib2.urlopen(
            'http://json-time.appspot.com/time.json')

json_data = json.load(data)
pprint(json_data)

print json_data['datetime']
```

lab_b_example_json_time.py

```
{u'datetime': u'Sun, 23 Jan 2011 17:15:32 +0000',
 u'error': False,
 u'hour': 17,
 u'minute': 15,
 u'second': 32,
 u'tz': u'UTC'}
```

```
Sun, 23 Jan 2011 17:15:32 +0000
```

questions?

Lab B (20)

why the obsession with addition and returning the time?

it’s a placeholder for *any* Python function you write. You can replace time.time() with nearly any function.

`git clone git://github.com/briandorsey/uwpython_web.git`

Lab B

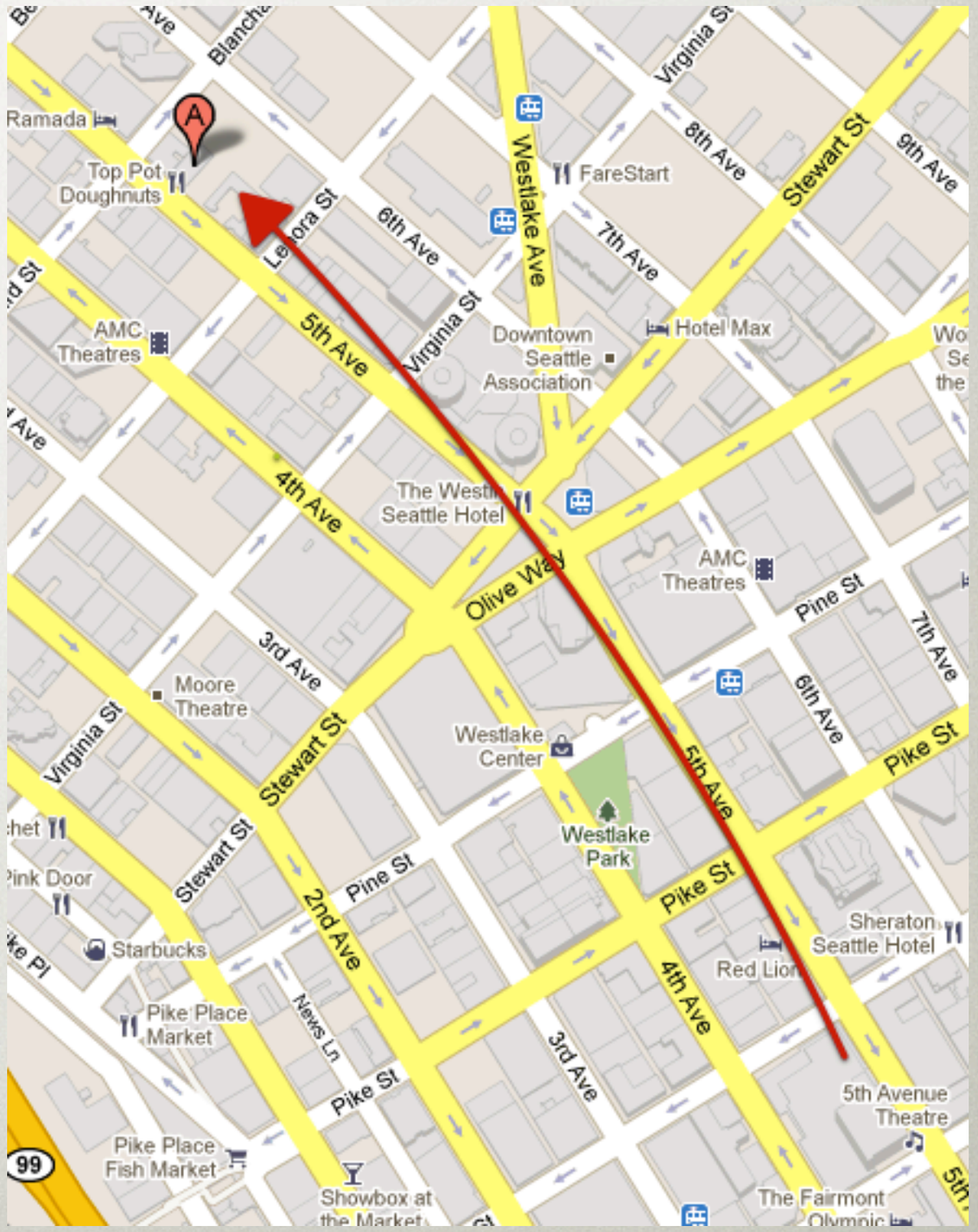
- update the echo server example
- write a **`time()`** function which returns the current time, call it from client
- write an **`add()`** function which adds two numbers, call it from client
- add a **`google_utc()`** function which returns the **`"datetime"`** from: http://json-time.appspot.com/time.json

WRAPUP

Info

Office hours:
Sunday 2-5pm

Top Pot Donuts
2124 5th Ave
Seattle, WA 98121
206-728-1966



Assignment

Choose one of the following:

- Write a program which accesses information from two web APIs and combines the data.
- Find and follow each of the links on the course syllabus, download the content and save it into a folder by week. Choose useful filenames.

  Also: set aside time to experiment with the commands from the reading for next week.